Microsoft®
Windows™ 95

CURSO IBM DE PROGRAMACION

14

EN Windows 95

IBM

PLANETA DeAGOSTINI

Microsoft® Windows™ 95

CURSO IBM DE PROGRAMACION EN Windows 95

VOLUMEN 2 - UNIDAD 14

Director Gerente: José Mª Puig de la Bellacasa
Director Técnico: Francisco Valer Bañales
Director de Marketing: Javier Graupera Gassol
Director Editorial: Artur Klein Moreno
Directora de Software: Beatriz Cerrato Vidal

Dirección y coordinación: José Luis Becerril Gasco. De la Corporación IBM España, Licenciado en Ciencias Físicas, Director del Instituto de Inteligencia Artificial de IBM España.

Redactores y colaboradores: *José Ramón Dorronsoro Ibero.* Doctor en Ciencias Matemáticas, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid, Investigador del Instituto de Ingeniería del Conocimiento (UAM). *Vicente López Martínez.* Doctor en Ciencias Químicas, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Manuel Alfonseca Moreno.* Doctor Ingeniero de Telecomunicación y Licenciado en Informática, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Javier Garrido Salas.* Doctor en Ciencias Físicas, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Javier Martínez Rodríguez.* Doctor en Ciencias Físicas, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Santiago de Aréchaga Tarruell.* Ingeniero Superior de Informática, Investigador del Instituto de Ingeniería del Conocimiento (UAM). *Pedro Pascual Broncano.* Profesor Asociado del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Carlos Aguirre Maeso.* Licenciado en Ciencias Matemáticas, Profesor Asociado del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid, Investigador del Instituto de Ingeniería del Conocimiento (UAM).

Realización editorial: LH servicios informáticos, S.L.

Diseño gráfico: Esteoeste

© Multimedia Ediciones, S.A., Barcelona
ISBN Obra Completa: 84-88720-79-3
ISBN Fascículos: 84-88720-80-7
ISBN Volumen 2: 84-88720-82-3
Depósito Legal: B-33.798-97

Redacción y administración:
Aribau 198, 5ª planta - 08036 - Barcelona
Tel. (93) 306.78.90 - Fax (93) 306.78.97

Servicio de atención al cliente: Teléfono 902.101.324

Petición de números atrasados:
Teléfono 902.288.288
Apartado de correos nº 2.867, -08080- Barcelona

Servicio de consulta:
Teléfono 902.244.444 de lunes a viernes, de 10.00 a 14.30 y de 15.30 a 19.00 horas.
Fax (91) 661.67.08 durante las 24 horas del día.

Distribuye para España: MIDESA. Marco Ibérica de Ediciones, S.A., Carretera de Irún, km. 13,350, variante de Fuencarral - 28034 - Madrid

Venta en quioscos y librerías: Pida a su proveedor que le reserve sus ejemplares del curso. Adquiriéndolos todas las semanas en el mismo quiosco o librería facilitará la distribución y obtendrá un mejor servicio. El editor se reserva el derecho de modificar el precio de venta de los componentes de la colección en el transcurso de la misma, si las circunstancias del mercado así lo exigieran.

Composición del Curso: El Curso IBM de Programación se publica en forma de 52 unidades de aparición semanal y 52 CD-ROM. Las 52 unidades se encuadernan en 5 volúmenes. Con la unidad que completa cada volumen se ponen a la venta las tapas para su encuadernación.

Multimedia Ediciones garantiza la publicación de todos los elementos que componen esta obra.

© Editorial Planeta Argentina S.A.I.C.
Independencia, 1668 - Capital Federal
Teléfono atención al cliente: 482-0655 y 482-0657
Distribuye en capital: Huesca-Sanabria. En interior: D.G.P.
© Editorial Planeta Mexicana S.A. de C.V.
Avda. Insurgentes # 1162 - México D.F.
Teléfono de atención al cliente: 575-1348
© Editorial Planeta Colombiana S.A.
Carrera 68 A, nº 25-55 - Santafé de Bogotá
Teléfono de atención al cliente: 9800-15315

Imprime: Cayfosa. Carretera de Caldes, km. 3, Santa Perpetua de Mogoda (Barcelona).

INSTALACION/DESINSTALACION DEL SOFTWARE: Para la correcta instalación y desinstalación del CD-ROM que se entrega en esta unidad, véase el libreto que acompaña a cada CD-ROM.

Marcas comerciales registradas: En esta publicación son marcas registradas de sus respectivas compañías: Delphi Desktop, dBase, Turbo Pascal, Turbo C++, Turbo Assembler, Turbo Debugger, Borland Intrabuilder, Paradox, Visual dBase y BRIEF (Borland International, Inc.). Lotus, Lotus 123 y SmartIcons (Lotus Development Corporation, filial de IBM). Microsoft DOS, Microsoft Windows, Microsoft Windows 95, Excel, Visual Basic, Access y FoxPro (Microsoft Corporation). Netscape Navigator Gold (Netscape Communications Corporation). Java (Sun Microsystems, Inc.). IBM (International Business Machines).

UNIDAD 14

CURSO DE PROGRAMACIÓN VISUAL: TURBO PASCAL (1)

Fig. 1. Blaise Pascal, eminente científico francés del siglo XVII que, además de matemático y físico, se distinguió también como filósofo y teólogo.

FUNDAMENTOS DE TURBO PASCAL

En esta unidad se comenzará el estudio del lenguaje de programación Turbo Pascal, que resulta especialmente adecuado para aprender los conceptos básicos que están detrás de todo lenguaje moderno de programación visual. Más adelante se estudiará el Turbo C++. Si bien este último es el más empleado para el desarrollo de aplicaciones destinadas a ordenadores personales, será necesario estudiar primero Pascal pues, además de utilizarse también para aquéllas, presenta la ventaja de una mayor facilidad para el aprendizaje de las ideas básicas de programación.

INSTALACIÓN

El proceso de instalación de la unidad es muy sencillo y sigue los mismos pasos de entregas anteriores. Insertar el CD con el programa en el lector del equipo, abrir el menú Inicio y activar la opción Ejecutar. Suponiendo que el lector CD es la unidad D: del ordenador, teclear: D:\INSTALAR.

Si el ordenador tiene activada la opción AutoRun de Windows 95, aparecerá la ventana de instalación sin necesidad de teclear ninguna orden. Entonces, únicamente hay que elegir el disco donde se desee ubicar Turbo Pascal y sus componentes. El programa de instalación copiará desde el CD los archivos necesarios, que se dispondrán en el directorio \TPW del disco elegido.

LOS LENGUAJES DE PROGRAMACIÓN

La información que puede procesar un ordenador es distinta a la que

puede procesar un ser humano, diferenciándose también el modo en que ambos lo hacen. La gran distancia que separa esos dos mundos marca el peculiar modo de comunicarse el hombre con la máquina. Podrá existir entre ellos algo distinto al teclado o el ratón, como por ejemplo un micrófono, pero lo que será extraño es que la máquina comprenda las expresiones habituales del hombre, que sea capaz de entender entre todas las interpretaciones de una frase la más adecuada en cada momento. La explicación es que el ordenador no entiende, por lo general, el lenguaje llamado natural, el que se utiliza todos los días para la comunicación. Y si hay que comparar a la máquina con un animal, en cuanto a su capacidad de comprensión, siempre se acabará utilizando la referencia al ser más tonto que cada uno imagine.

Aparte de los matices que aportan los distintos idiomas, el lenguaje natural permite expresar conceptos y situaciones muy complejas. Para ello, hace uso de la multiplicidad de significados que una misma palabra puede adquirir dependiendo del contexto.

Con el ordenador se deben utilizar órdenes precisas, sin ambigüedades, a las que se da el nombre de comandos. A veces, se puede sustituir teclear el comando por elegir una opción con el ratón entre las disponibles en la pantalla. Siempre se desencadena un proceso que está asociado, sin ambigüedad, al comando o a la opción elegida. Actualmente se investiga para hacer más fluida la comunicación entre el ordenador y el usuario, procurando un acercamiento cada vez mayor entre el lenguaje natural y los que pueda interpretar aquél. Pero, por ahora, habrá que conformarse con programar el ordenador indicándole secuencias de órdenes precisas.

Para comprender fácilmente los conceptos fundamentales de programación resulta conveniente tener siempre presente un modelo esquemático del ordenador; aquello a lo que básicamente se reduce toda la máquina en cuanto a "inteligencia" se refiere. Los elementos fundamentales del esquema son la Unidad Central de Proceso (UCP) y la memoria. La primera es la parte esencial del ordenador, que se encarga de dirigir el tráfico de información. La memoria es donde se almacena aquélla; esta memoria puede ser volátil (memoria RAM) o duradera (grabaciones en dispositivos magnéticos, como discos o disquetes).

Cualquier dispositivo de captura de datos (teclado, ratón...) se comunica con la UCP, que es además la encargada de ordenar la salida de datos (videoterminal, impresora...). La UCP consta de dos partes, la Unidad de Control (UC), que dirige el tráfico de información, y la Unidad Aritmético-Lógica (UAL), que puede realizar operaciones muy sencillas como sumas, restas y comparaciones. Con esto se termina lo que sabe hacer la unidad central de procesos, el resto es todo programación.

Después de lo dicho puede pensarse que el ordenador es una máquina poco inteligente. Pues bien, lo será dependiendo de su memoria, ya que con ella y rapidez podrá paliar el déficit de inteligencia. En la memoria estarán almacenadas todas las operaciones que deberá realizar la UCP y los datos que pueda utilizar. Por lo demás, todo es cuestión de desmenuzar las tareas complejas en otras más sencillas hasta llegar a las cuatro cosas que sabe hacer la UAL, y luego llevarlas a cabo muy rápidamente. No importa que alguien tenga que sumar ocho veces once para saber que 11 x 8 son 88 si tarda menos que otro que sabe multiplicar.

Mientras se tenga el ordenador trabajando, la UCP estará trayendo y llevando datos a la memoria y, de vez en cuando, ordenando a la UAL que haga algún cálculo. La rapidez con que todo este proceso se lleva a cabo hará parecer que la máquina sabe leer, dibujar y realizar otros muchos trabajos.

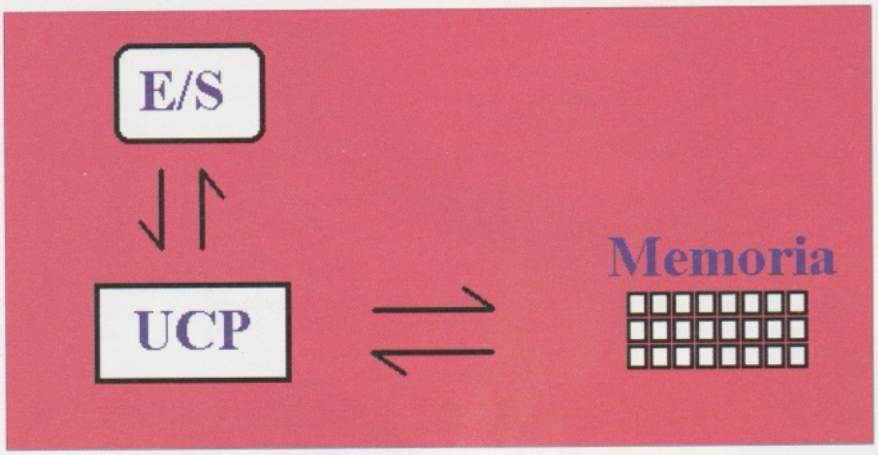


Fig. 2. El funcionamiento del ordenador se puede resumir como el tráfico de datos entre el exterior y la memoria, dirigido todo por la Unidad Central de Proceso.

Fig. 3. Desde la Unidad Central de Proceso se derivan las operaciones matemáticas y lógicas a la UAL.

De lo dicho se deduce que el ordenador no comprende conceptos abstractos y atemporales. Sólo es una máquina que puede recorrer, ordenadamente, la sucesión de estados que se le tracen. Por ello, no debe el usuario confundirse cuando encuentre en un listado de un programa de ordenador, por ejemplo en lenguaje C, una expresión como ésta:

```
x = 100 + 1;
```

La expresión no se refiere, como podría pensarse, a la equivalencia entre un número x y la suma de 100 y 1. Se trata de algo muy distinto, de una instrucción de **asignación**. En el lenguaje Pascal se marcan más las diferencias y la asignación equivalente se escribe:

```
x := 100 + 1;
```

con el símbolo := indicando que el resultado de la suma se ha de almacenar en x.

Mientras que una expresión matemática se puede referir a un concepto abstracto, la instrucción de asignación indicará una secuencia de órdenes muy precisa: el número 100 y el número 1 han de sumarse, almacenando el resultado en el lugar de memoria que identifica el nombre x. En el esquema del ordenador se estará indicando a la UCP que traiga de la memoria esas dos cantidades y que, después de sumarlas, el resultado lo vuelva a llevar a algún lugar de aquélla. Se puede dejar al ordenador que elija el lugar concreto, pero con el nombre x se sabrá siempre qué lugar es ese y así será posible hacer referencia a él en fases posteriores del programa.

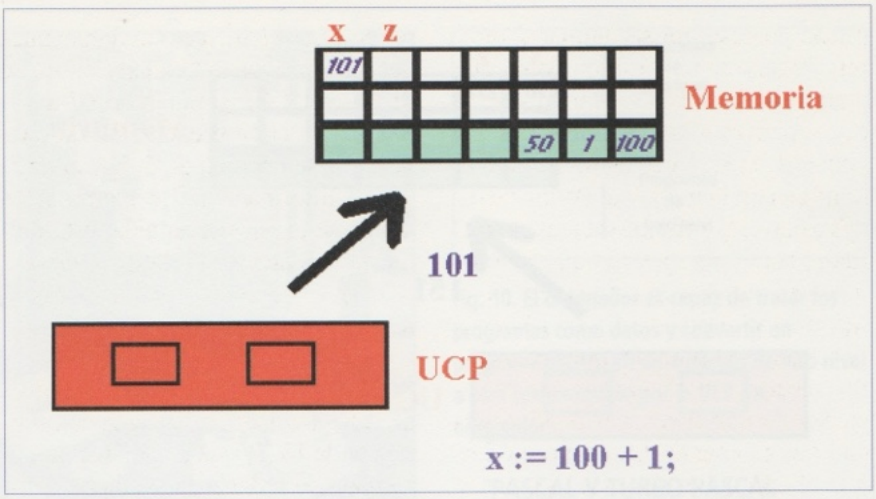


Fig. 5. Para finalizar la ejecución de x := 100 + 1; después de sumados los números en la UAL, el resultado se devuelve a la memoria y se guarda en la posición reservada para la variable x.

el ordenador, añadiéndole una nueva instrucción al programa. El nuevo será:

```
x := 100 + 1;
z := x + 50;
```

lo que significa que el programa trazado para la UCP empieza por sumar los números 100 y 1 y el resultado ha de almacenarlo en el lugar de memoria identificado con la letra x. Después, ha de sumar 50 al número que

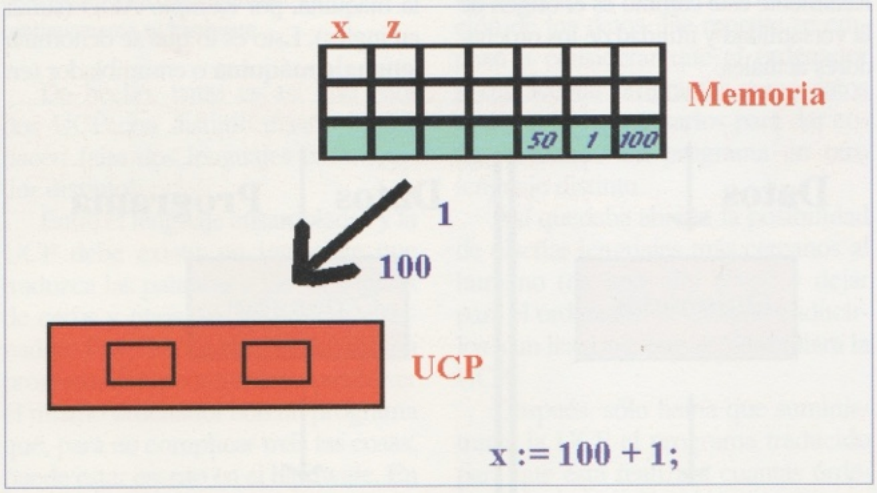


Fig. 4. Para realizar la asignación x:= 100 + 1; primero llegan a la UCP los datos 100 y 1 almacenados en la parte de la memoria reservada para guardar el código del programa.

Ahora se va a complicar un poco más el mandato que tendrá que seguir guardó anteriormente en ese lugar referido con la x, y finalmente, el nuevo resultado volverlo a guardar en otro lugar de memoria que le parezca oportuno y que estará identificado con la letra z.

Una vez finalizadas todas las operaciones que se le han indicado, la UCP habrá guardado el número 151 en el lugar de memoria que haya elegido para z. Es muy importante comprender que hay que seguir un orden muy rígido al proporcionar las instrucciones, ya que el ordenador interpretará siempre los programas en una sucesión muy precisa: de arriba a abajo y de izquierda a derecha. Conviene recordar que el programa es como el manual de instrucciones para manejar una máquina y el resultado final deseado únicamente se conseguirá si se siguen estas instrucciones.

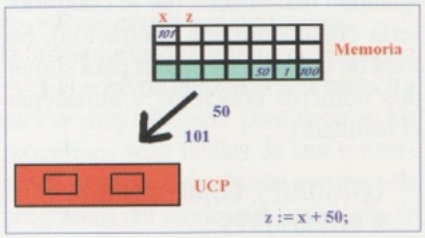


Fig. 6. Para realizar la asignación z := x + 50; primero hay que recuperar de la memoria el dato almacenado en el lugar reservado para la variable x, y después la constante 50 almacenada en la zona reservada para el código del programa.

Ya ha llegado el momento de poder presentar un tipo de instrucción que aparece muy frecuentemente en los programas de ordenador: la de incremento. En Pascal sería, por ejemplo,

```
x := x + 2;
```

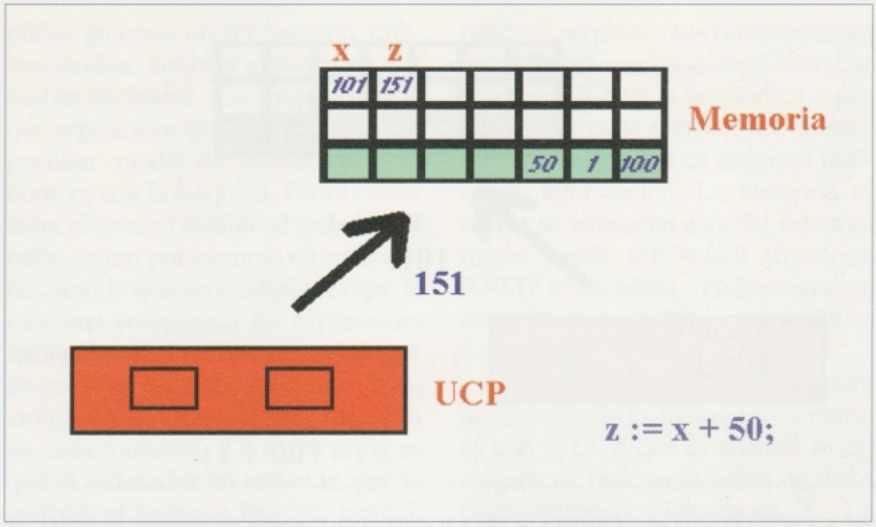


Fig. 7. Acaba de realizarse la instrucción z := x + 50; cuando se almacena en el lugar de la memoria reservado para la variable z el resultado de la suma.

que ordena sumar 2 al dato almacenado en x, guardando de nuevo la cantidad incrementada en el lugar identificado por x. Esta instrucción se encontrará formando parte de muchos programas y el aspecto que adquirirá depende del lenguaje de programación. Por ejemplo, en el lenguaje C, la misma orden dada a la UCP se escribiría:

```
x = x + 2;
```

que ya se sabe que no tiene nada que ver con una expresión abstracta de igualdad matemática (de lo contrario, sería claramente una aberración). Se trata de una orden escueta para sumar dos números concretos y almacenar el resultado.

LENGUAJES, COMPILADORES E INTÉRPRETES

El modo en que se suministran las órdenes a la UCP ha variado mucho desde la época de los ordenadores primitivos. Éstos se programaban cambiando directamente de posición los cables que unían las partes del ordenador.

Eran instrumentos que hacían, generalmente, cálculos matemáticos. Realizaban una secuencia de operaciones, siempre la misma, con los datos de entrada. Para elegir otra secuencia distinta, otro programa, había que cambiar cables de sitio. El matemático Von Neumann, uno de los padres insignes de la informática, tuvo la brillante idea de utilizar parte de los datos de entrada como programa del cálculo que debía seguir el ordenador con el resto de los datos. Con este enfoque, entonces revolucionario, se podía dejar fijo el hardware y variar la utilización de la máquina simplemente modificando los datos que se le suministraban. Realmente este cambio es el origen de la versatilidad y utilidad de los ordenadores actuales.

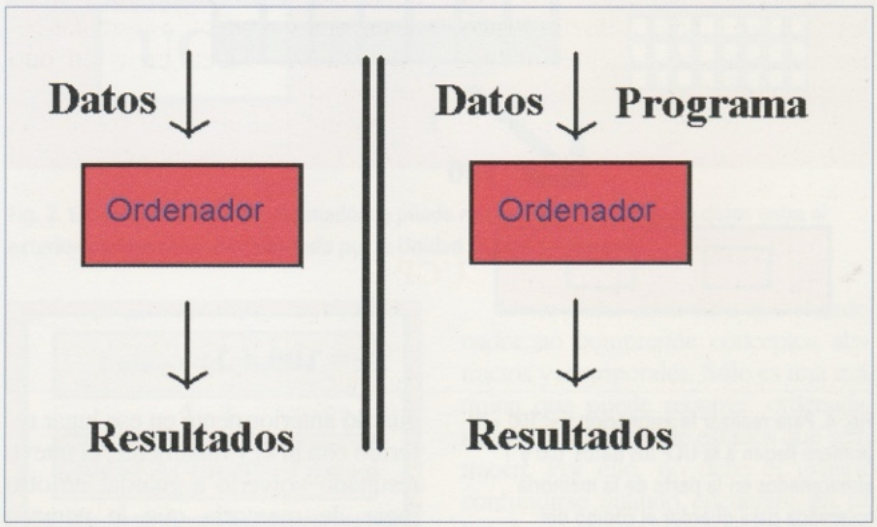


Fig. 8. Los ordenadores actuales deben su versatilidad a la posibilidad de recibir datos de entrada que son el programa que ha de dirigir sus tareas.

Está claro que los datos que había que suministrar entonces al ordenador eran los que directamente entendía la UCP, es decir, secuencias de ceros y unos. En el primitivo ordenador MARK I se utilizaban secuencias de tripletes de perforaciones en un cinta. Afortunadamente, resultó muy sencillo darse cuenta de que la "conversación" podía hacerse un poco más humana utilizando mnemotécnicos para las secuencias de ceros y unos que constituían la parte de los datos que eran el "programa".

Fue un alivio pasar de programar el ordenador con "frases" como:

```
0010 0100
0100 0101
0011 0110
```

a frases del tipo:

```
LOAD  X
ADD   Y
STORE Z
```

que no son más que una interpretación de las secuencias de 0 y 1, por ejemplo 0100, transformada en recordatorios de lo que esa secuencia ordena hacer a la máquina, por ejemplo ADD (suma en inglés). Esto es lo que se denomina **lenguaje máquina** o ensamblador (en inglés, Assembler). A este tipo de lenguajes se les denomina de bajo nivel porque están muy próximos al diseño de la UCP.

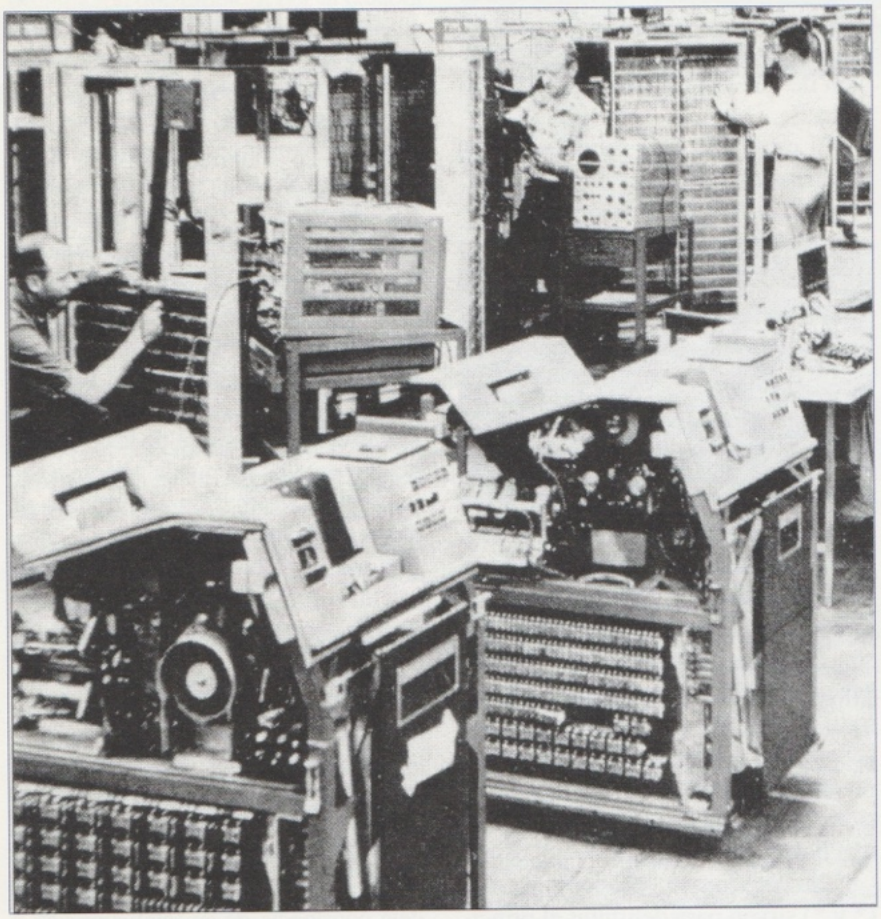


Fig. 9. En un principio, para programar un ordenador había que modificar continuamente su hardware.

De hecho, tanto es así que para dos UCP con distinto diseño lógico hacen falta dos lenguajes ensamblador distintos.

Entre el lenguaje ensamblador y la UCP debe existir un intérprete que traduzca las palabras a las secuencias de ceros y unos que asimila el ordenador. Este intérprete puede ser un programador, pero también puede ser el mismo ordenador con un programa que, para no complicar más las cosas, puede estar escrito en el hardware. En este planteamiento ya está implícita una revolución que incomodó a no pocos programadores de antaño, pero que posteriormente facilitó el manejo a otros muchos.

Hasta entonces se consideraba que el ordenador aceptaba datos (generalmente números) y los procesaba para encontrar resultados (también números en general), y los programas se encargaban de dirigir esa manipulación de los datos. De repente se empezó a considerar que el ordenador podía aceptar programas como datos de entrada y procesarlos para dar como resultado un programa en otro lenguaje distinto.

Así quedaba abierta la posibilidad de diseñar lenguajes más cercanos al humano (de más alto nivel) y dejar para el ordenador la tarea de traducirlos a un lenguaje que comprendiera la UCP.

Después, sólo había que suministrar a la UCP el programa traducido para que ésta realizara cuantas órdenes se habían deseado.

Al programa que se encarga de la traducción se le denomina compilador. No importa cómo esté programado el compilador ni en qué lenguaje se haya escrito ya que, una vez que éste sea comprensible para la UCP, también lo será cualquier programa escrito en el lenguaje que "compila".

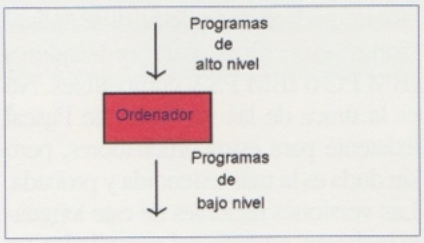


Fig. 10. El ordenador es capaz de tratar los programas como datos y convertir un programa escrito en un lenguaje de alto nivel a otro comprensible por la UCP del ordenador.

PASCAL Y TURBO PASCAL

Pascal es un lenguaje de programación relativamente moderno, desarrollado por Niklaus Wirth y su grupo de Zurich en 1971. Se trata de un lenguaje de propósito general, esto es, que se puede emplear para construir todo tipo de aplicaciones.

En la práctica también quiere decir que se trata de un lenguaje no diseñado para desarrollar ningún tipo específico de aplicaciones. Pero Pascal es especialmente útil para la enseñanza de "buenos modos de programación" debido a los siguientes motivos:

- Posee unas reglas sencillas de sintaxis.
- Es un lenguaje muy estructurado.
- Realiza una comprobación exhaustiva de los tipos de datos.

El hecho de que tenga una estructuración muy definida permite que los programas sean fáciles de leer e interpretar, y además facilita la escritura de programas del modo que hoy en día se estima correcto.

El compilador de Pascal es relativamente sencillo de realizar, por lo que se ha extendido a muchos tipos de plataformas, desde los ordenadores personales a los grandes de tipo corporativo. Cuando una aplicación se escribe en Pascal estándar puede compilarse en cualquier máquina en la que exista un compilador de Pascal, que son la mayoría.

Turbo Pascal es una derivación de la anterior que incluye, además de las instrucciones del lenguaje Pascal estándar, una serie de instrucciones o fun-

ciones que permiten desarrollar aplicaciones específicas para ordenadores IBM PC o IBM PS y compatibles. No es la única de las versiones de Pascal existente para estos ordenadores, pero sin duda es la más extendida y probada. Las versiones recientes de este lenguaje incluyen, además del compilador, un entorno integrado de desarrollo (IDE) que permite, dentro de una misma herramienta, realizar todos los pasos necesarios para escribir el programa, editarlo y obtener el código máquina.

LOS PROGRAMAS DE PASCAL

Un programa de Pascal es un conjunto de instrucciones que siguen la sintaxis y la estructura del lenguaje. La estructura genérica es:

```
Program nombre;
   .   .   .
   instrucciones declarativas
   .   .
Begin
   .   .   .
   sentencias
   .   .   .
End.
```

Todo programa Pascal empieza con la palabra Program seguida de un nombre que elige el programador para identificar el programa. Los signos de puntuación no son arbitrarios, sino que forman parte de la sintaxis del lenguaje y permiten eliminar ambigüedades en la interpretación de las órdenes que ha de analizar el compilador. La primera línea del programa acaba con el signo de puntuación ";" que es la forma habitual de finalizar un instrucción en Pascal.

Después de la identificación del programa se han de situar las instrucciones declarativas de aquél, que sirven para especificar el significado de los términos que se utilizarán en la parte del programa que ha de traducirse finalmente al lenguaje que comprende la UCP. A continuación, ha de aparecer el bloque que contiene estas instrucciones, y debe encabezarse con Begin, terminando con End y un punto. El punto final es eso: una indicación de que el programa se ha terminado.

El programa más pequeño y más inútil que cumple las reglas de estructuración de Pascal es el siguiente:

```
Program nulo ;
 { Ejemplo de la estructura más
simple de un programa PASCAL }
Begin
(* No hace falta ninguna instrucción
para no hacer nada *)
End.
```

En la parte reservada a declaraciones no se incluye nada, pues nada se necesita declarar. Todos los símbolos que se encuentren entre las llaves { } son comentarios que sirven para hacer más legible el programa. Es decir, todo lo que el compilador encuentre después de la llave {, y hasta que aparezca su compañera de cierre }, considerará que no son órdenes Pascal que hayan de traducirse.

También los símbolos compuestos (* y *) sirven para delimitar el principio y fin de un comentario. El compilador sólo da por finalizado aquél que se inicia con uno de los dos símbolos posibles cuando encuentra, exactamente, la pareja del inicial. Así, combinando los dos pares de símbolos, es posible escribir un comentario dentro de otro, de modo que posteriormente se pueda eliminar el más externo.

Un programa debe ilustrarse con abundantes comentarios. Sirven para recordar al autor por qué eligió una orden concreta para llevar a cabo un algoritmo. Sin los comentarios, los programas tienden a ser ilegibles a medida que pasa el tiempo transcurrido desde que se escribieron. Son imprescindibles para poder mantener actualizados los programas. Especialmente, si es posible que sea una persona distinta al autor original quien tenga que modificarlos en el futuro.

Las palabras Program, Begin, End y otras que se verán más adelante son identificadores que permiten al compilador Pascal interpretar el programa y que no se pueden utilizar con otros fines. Para todas ellas se usa el término de palabras reservadas.

El compilador interpreta las mayúsculas o minúsculas por igual y, por tanto, pueden usarse indistintamente. Esta libertad de elección debe utilizarse para aumentar la legibilidad de los programas. Por ejemplo, se pueden empezar siempre con mayúscula las palabras reservadas y así será más fácil detectar las estructuras del programa. También suele escribirse con mayúsculas el nombre de las constantes cuyo valor no puede alterarse durante el programa. La elección de un estilo u otro de escritura es una cuestión personal, pero es importante que sea uniforme a lo largo del programa y sirva para aumentar su claridad.

EL PRIMER PROGRAMA

En casi todos los cursos de programación el primer programa que escribe el alumno consiste en hacer que el ordenador le salude. En esta unidad se respetará la tradición haciendo que la UCP deje constancia de su total sumisión a las órdenes de quien sepa programarla. El programa es simplemente:

```
Program Saludo ;
   Uses WinCrt ;
   Begin
   {1}   WriteLn( ' Hola Luis ' ) ;
   End.
```

y sigue la normas sintácticas del lenguaje Pascal. Empieza por el nombre del programa, "Saludo", y sólo contiene una instrucción que se ha numerado con el 1 entre símbolos de comentario. La instrucción es una función básica de intercambio de datos. Se trata de la función WriteLn() que sirve para escribir en la pantalla todo aquello que se especifique en el argumento. En este caso es un conjunto de caracteres que se delimitan con la comilla sencilla ' . Aunque aquí se ha escrito Luis, cada lector puede cambiar este nombre por el suyo propio.

En la parte declarativa del programa, antes de Begin, se incluye una

declaración que empieza por la palabra reservada Uses (utiliza). Se trata de una declaración que indica al compilador que, para interpretar cuanto sigue a Begin, necesita la información que existe en una unidad externa denominada WinCrt. Por unidad externa se entiende otro programa, o librería de programas, que se encuentra ya compilado y disponible en algún lugar accesible para el compilador o el montador. En este caso concreto, la unidad WinCrt es la que almacena todo el código que hace posible la gestión del terminal de acuerdo con los estándares de Windows. Al incluir esta unidad, el saludo se escribirá en una ventana como todas las de Windows, que se abrirá para poder mostrar cuanto se escriba con la función WriteLn.

El programa "Saludo", por su sencillez, va a permitir ilustrar los pasos que hay que seguir para convertirlo en órdenes de la UCP sin complicaciones adicionales debidas a la sintaxis del programa. Este primer ejercicio tiene como objetivo generar un fichero ejecutable saludo.exe tal que, al ejecutarlo desde Windows, aparezca en la pantalla del ordenador la ventana del programa con el saludo al programador.

LOS PRIMEROS PASOS CON TPW

Una vez instalada la primera entrega de Turbo Pascal para Windows, se encontrará en el menú de Inicio el grupo TPW, que contiene las herramientas que forman el Entorno Integrado de Desarrollo de Turbo Pascal para Windows 95. Para empezar a trabajar con él, hacer doble clic sobre el icono de TPW.

La pantalla inicial que aparecerá en el ordenador es la misma que puede verse en la figura 12.

La estructura de este entorno es común a muchas otras aplicaciones que ya se han estudiado.

En la parte superior se encuentra la barra con los comandos que gobiernan TPW y, justo debajo, la barra de herramientas que permite ejecutar los comandos más comunes directamente con el ratón.

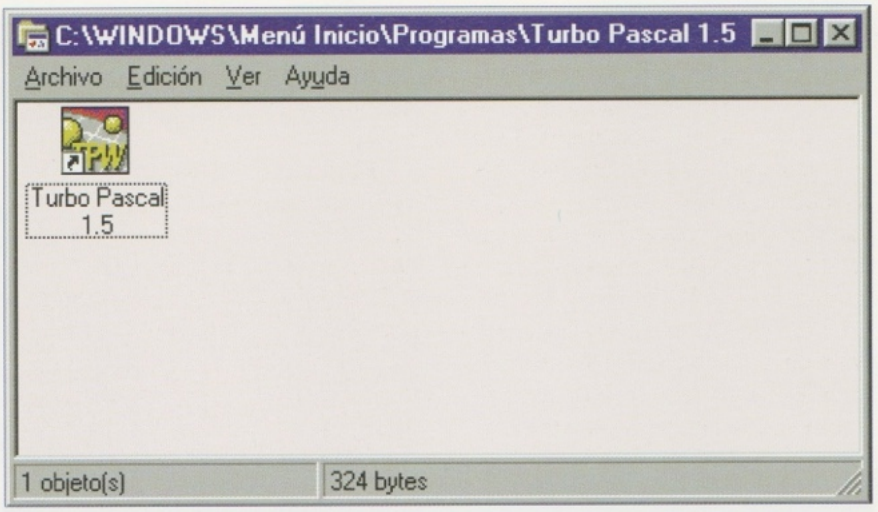


Fig. 11. Grupo de Turbo Pascal para Windows 95 con el icono del entorno integrado de desarrollo.

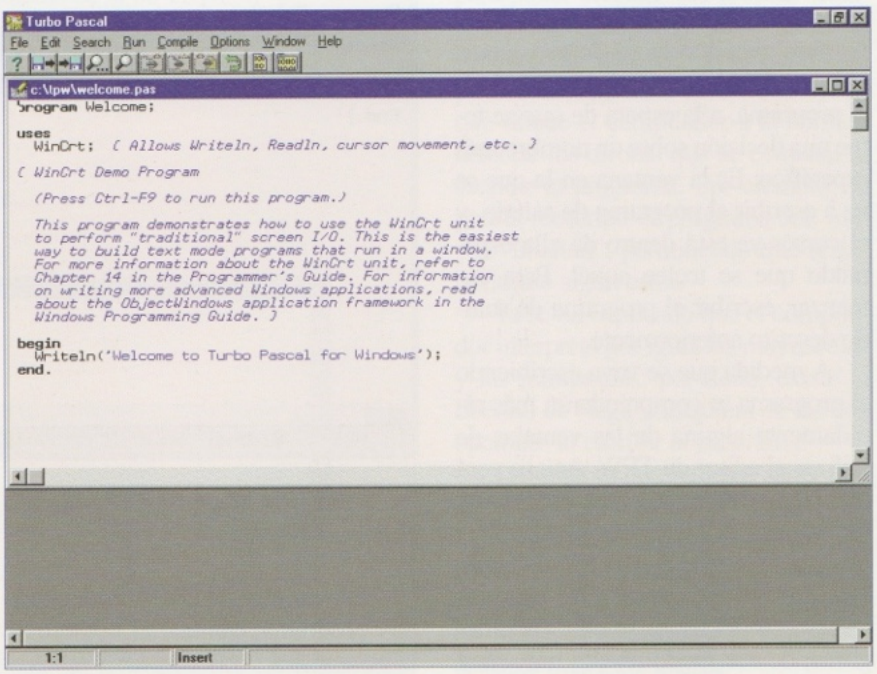
El centro de la pantalla lo ocupa una ventana que contiene un programa Pascal. Se trata del programa "welcome", uno de bienvenida que ha quedado instalado en el archivo "welcome .pas" del directorio TPW, donde se almacena el resto de Turbo Pascal para Windows.

El programa aparece dentro de una ventana que comparte sus propiedades con las de Windows. Se puede desplazar, agrandar y también cerrar.

Fig. 12. Pantalla inicial de Turbo Pascal para Windows.

Por ahora es eso con lo que se debe practicar: cerrar la ventana que contiene el programa y dejar el espacio de trabajo vacío.

El ejercicio inicial consistirá en escribir el programa de saludo, una versión más personal del "Welcome". El objetivo es crear los archivos "saludo .pas", en el que se guardará el programa, y "saludo.exe", en el que se guardará el ejecutable que, al ser invocado desde Windows, producirá el saludo. A todos los archivos que contienen programas en Pascal se les suele dar la extensión .pas.



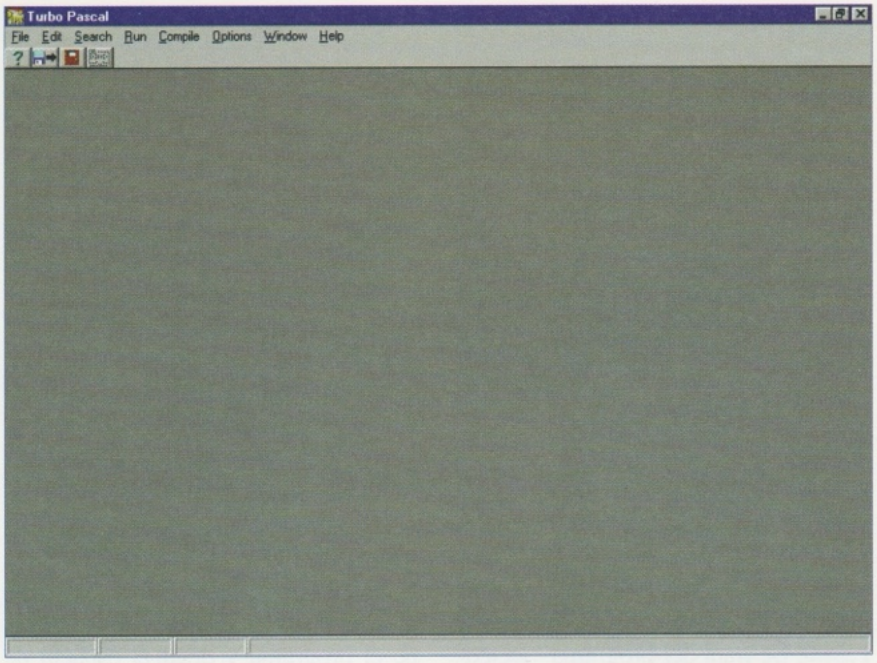


Fig. 13. Espacio de trabajo de TPW donde se abrirán ventanas que contienen programa.

El saludo.pas será un archivo ASCII cuyo contenido se podrá modificar o crear con cualquier editor de textos ASCII, pero resultará más conveniente trabajar con estos archivos desde dentro del entorno integrado de desarrollo de TPW, que también incluye un editor.

En la barra de menús seleccionar la opción **File | New (Archivo | Nuevo)** y aparecerá en pantalla una ventana llamada "noname00.pas", que es el nombre genérico tanto de la ventana como del archivo donde se guardaría el programa, a la espera de que se tome una decisión sobre un nombre más específico. Es la ventana en la que se va a escribir el programa de saludo, y el cursor ya está dentro de ella esperando que se teclee aquél. Para comenzar, escribir el programa de saludo descrito anteriormente.

A medida que se vaya escribiendo el programa se comprenderán más rápidamente alguna de las ventajas de utilizar el editor de TPW. Así, se verá que interpreta las palabras que se van introduciendo y para escribirlas utiliza un tipo de letra que depende de aquéllas. Por ejemplo, resalta las palabras reservadas del lenguaje utilizando negritas; de ese modo resulta muy fácil evitar errores ya que, si al acabar de escribir una palabra reservada no cambia a negrita, es debido a que el usuario se ha equivocado al teclearla.

Una vez acabado de escribir el primer programa, cambiar el nombre de la ventana por "Saludo.pas" y aprovechar para guardar el archivo en el disco. Utilizar para ello la opción **File | Save as... (Archivo | Guardar como...)**. Comprobar que, tras quedar guardado el programa en el archivo, aparezca en la ventana no sólo el nombre, sino también la ruta del directorio en el que se almacenó. En este caso, si no se ha elegido una ruta especial, aparecerá C:\tpw\saludo.pas.

Para convertir el contenido de la ventana activa en código comprensible por la UCP hay que realizar varios pasos. Elegir la opción **Compile | Compile** (Compilar) o pulsar la combinación de teclas **Alt-F9**. La interpretación de este programa tan sencillo apenas llevará tiempo y casi inmediatamente aparecerá en pantalla un mensaje informando que la compilación del programa se ha efectuado sin contratiempos. Esto será así, claro está, si no se ha cometido ningún error al escribir el programa, es decir, si se han escrito bien todas las palabras que lo componen y no se ha modificado ningún signo de puntuación.

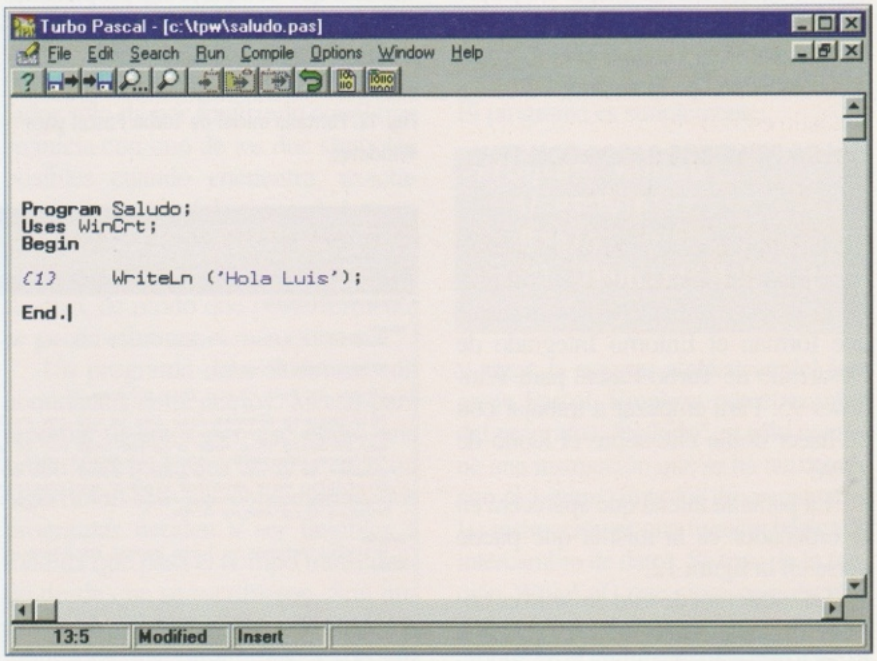


Fig. 14. Programa saludo. El editor modifica la presentación de las palabras según el papel que jueguen en el lenguaje.

Más adelante se analizarán los errores que se hayan podido cometer y cómo responde a ellos el compilador.

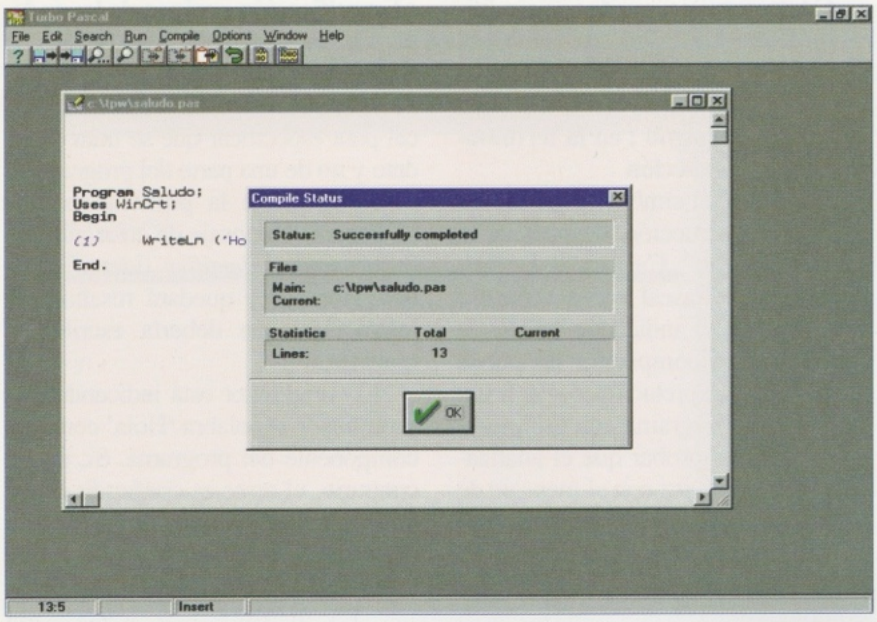


Fig. 15. Ventana que informa del resultado de la compilación que se ha realizado.

Si todo ha ido bien, TPW habrá guardado en el archivo "Saludo.exe" una versión en lenguaje comprensible por la UCP de las instrucciones contenidas en el programa compilado.

¿Se ha avanzado algo con esto? Pues bastante, ya que se ha permitido al compilador opinar sobre si el programa estaba bien escrito en el sentido de cumplir las normas sintácticas del lenguaje Pascal; además, lo que ha quedado escrito en "saludo.exe" realizará la acción deseada en cuanto pase a la memoria del ordenador.

Para comprobarlo, se puede acceder al menú de Inicio de Windows 95 y elegir la opción Ejecutar para iniciar el programa. Pero esto también se puede lograr desde dentro del entorno integrado de desarrollo de TPW. Basta con elegir la opción **Run | Run** (Ejecutar) o pulsar la combinación de teclas **Ctrl-F9** para ver el efecto que produce el programa: aparece en primer plano la ventana de Windows que contiene el saludo. Se trata de una ventana de Windows completamente independiente de la de TPW.

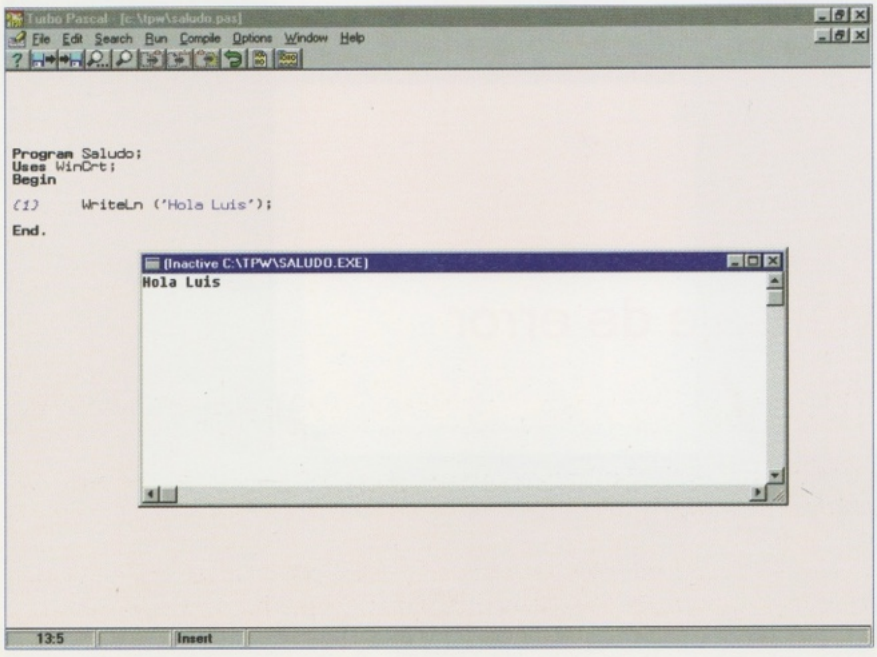


Fig. 16. El programa saludo se ha ejecutado y cumple la tarea encomendada.

En este caso quiere decir que no se debe olvidar cerrarla una vez acabado de leer el mensaje pues, de lo contrario, ocupará memoria del ordenador durante la sesión de Windows. Comprobar que esto es así pulsando **Alt-Tab** para ver las aplicaciones activas, o bien **Ctrl-Alt-Supr** para ver la lista de tareas y aprovechar para cerrarla desde la lista.

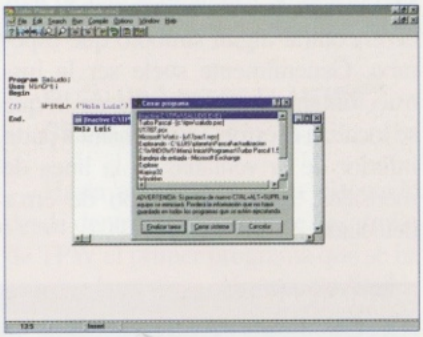


Fig. 17. El programa de nuestro ejemplo es una más de las tareas que se ejecutan en la sesión de Windows.

LOS MENSAJES DE ERROR

Probablemente el usuario desee saber más sobre la sintaxis del lenguaje para poder efectuar sus propias variaciones del programa. Pero antes es conveniente que siga practicando un poco con él para así conocer la ayuda que ofrece el compilador a la hora de detectar los errores que se cometan al escribir el programa. Por ejemplo, se podría reescribir el programa con mayor "libertad", permitiendo que se introdujese algún error.

Ya se ha explicado que el compilador interpreta por igual las mayúsculas y las minúsculas; por tanto, existe la libertad de elegir cualquiera de las dos opciones al escribir el programa.

También es conveniente saber que aquél interpreta igual un espacio en blanco que muchos, y que los cambios de línea son irrelevantes; así, el usuario

puede dejar cuantos espacios desee en el lugar en el que, al menos, hay uno. Por lo demás, salvo el cambio del nombre por el propio, no se debe modificar ni omitir ningún carácter o signo de puntuación respecto al modelo dado. A continuación, la propuesta es que se cometan una serie de equivocaciones controladas para ver la respuesta que ofrece TPW. Cuando el compilador analiza una instrucción y encuentra que no sigue las normas sintácticas del lenguaje Pascal, detiene la compilación del programa y avisa del error. De entrada, resalta la línea donde deja de encontrar algún símbolo que esperaba. Generalmente suele ser la instrucción siguiente a aquella en la que se localiza el error. También en la parte inferior de la ventana, en la línea de mensajes, informa del tipo de error detectado.

Como ejercicio, puede irse modificando el programa para que produzca, uno a uno, los errores que se comentarán a continuación.

• Omisión del signo ; en la terminación de la instrucción

No incluir el símbolo ; al final de la primera instrucción, después de la palabra "saludo". Casi todas las instrucciones de Pascal acaban con ese símbolo. Sin el indicador de fin de instrucción, el compilador no puede realizar la interpretación de la frase. Compilar el programa con esa modificación y comprobar que el analizador sintáctico avisa con el mensaje de error número 85: ; *expected* (; esperado), y resalta la línea donde se ha detenido la compilación. Añadir el símbolo omitido y proceder de nuevo a la compilación para comprobar que se ha arreglado el problema.

• Especificación errónea de las cadenas de caracteres

La frase 'Hola Luis' ha de encontrarse entre las comillas que utiliza Pascal para especificar que se trata de un dato y no de una parte del programa.

Si se omite la primera comilla, aparecerá el mensaje de error número 3: *Unknown identifier* (identificador desconocido) y quedará resaltada la instrucción que debería escribir el mensaje.

El compilador está indicando que no entiende la palabra 'Hola' como un componente del programa. Si, por el contrario, el error es confundir la comilla y utilizar la doble comilla ", se obtendrá el mensaje 5: *Syntax error* (error de sintaxis).

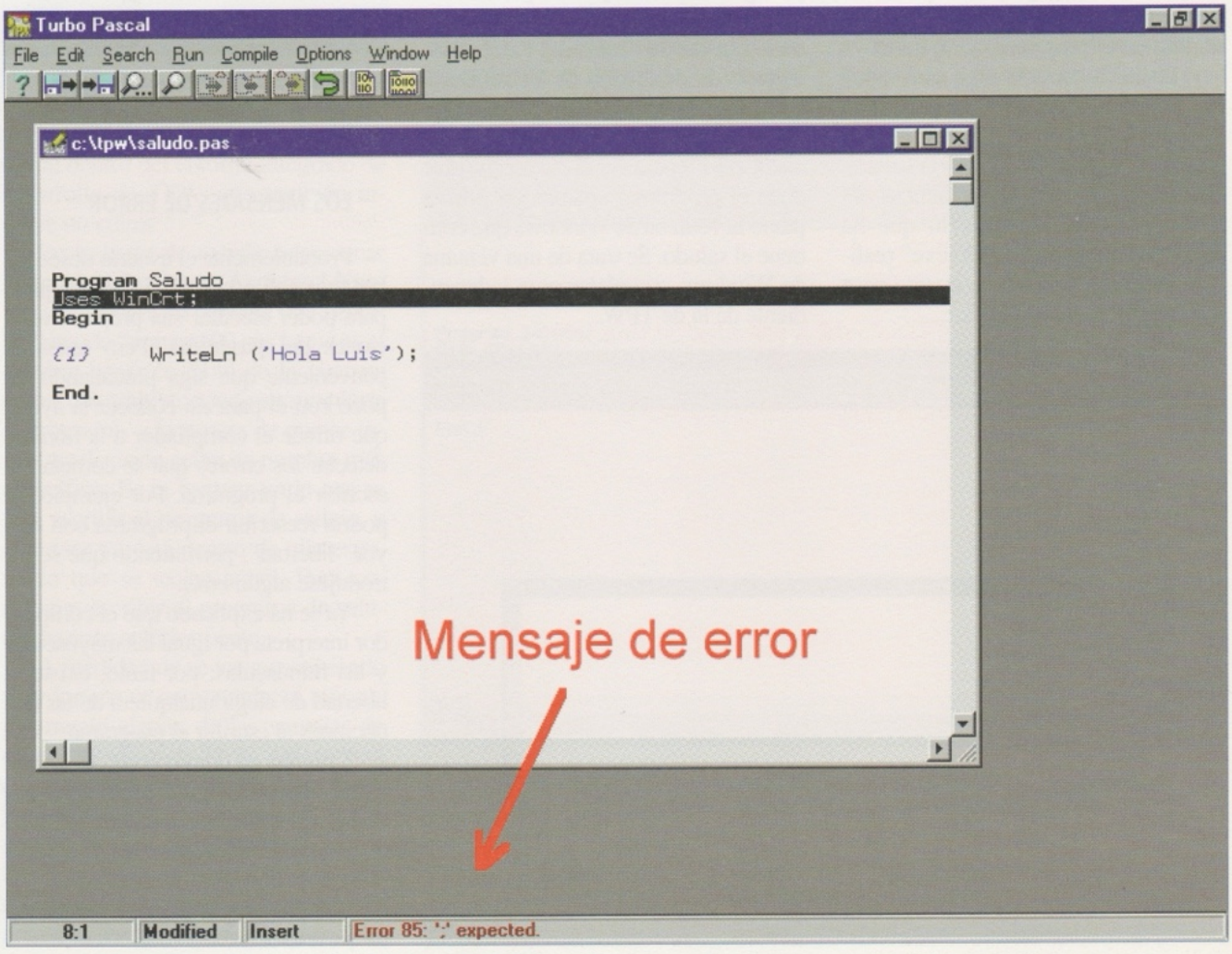


Fig. 18. Omisión del símbolo de punto y coma (;) al final de la primera instrucción.

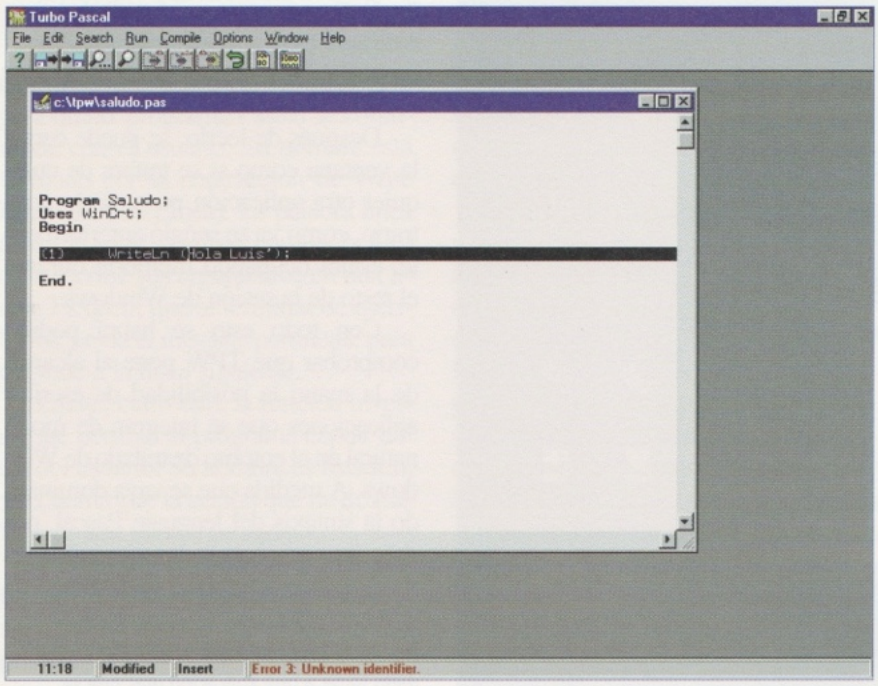
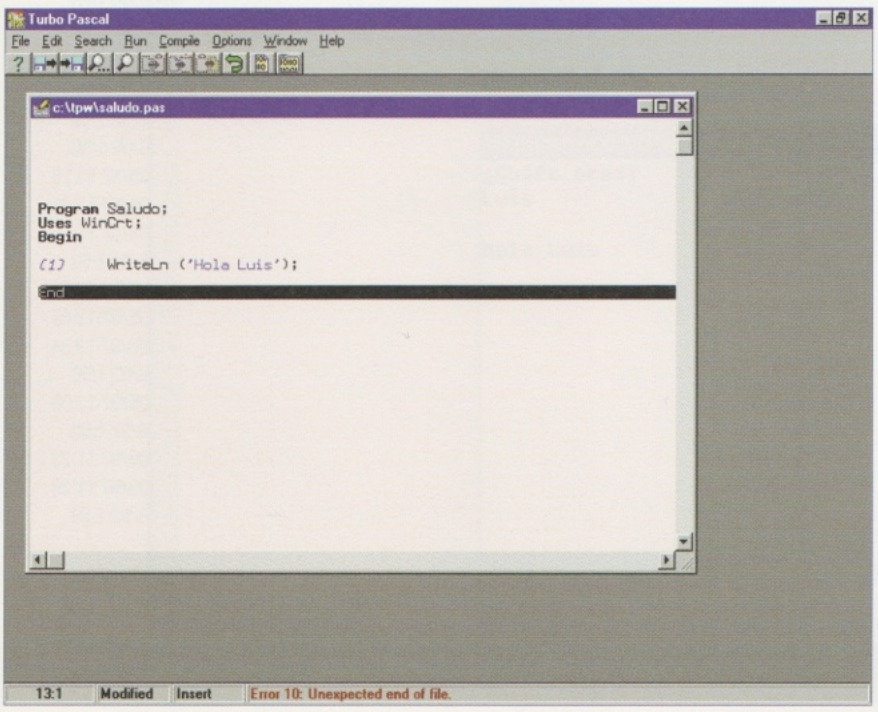


Fig. 19. Se ha omitido la comilla antes de empezar el mensaje de saludo.

• Omisión del punto final

La última palabra de un programa Pascal siempre es "End." y el punto final no debe olvidarse. Sirve para indicar al compilador que no se trata del final de una instrucción compleja, sino de todo el programa.

Si se omite este punto final, se reflejará en la pantalla el mensaje de error número 10: *Unexpected end of file* (fin de fichero inesperado), resaltándose la última línea del programa. El mensaje significa que se ha acabado el archivo antes de lo esperado, y a esa conclusión ha llegado el compilador porque no ha podido encontrar el "End." con el que termina todo programa.



Puede realizarse también el experimento de omitir el símbolo ; al final de la instrucción comentada con {1}. Se comprobará que en este caso no hay mensaje de error. Esto es así porque el símbolo de fin de instrucción no debe colocarse en aquellas que vayan seguidas por End. Esta palabra reservada ya es suficiente indicador de que la instrucción anterior ha terminado. Aunque parezca extraño a primera vista, no es que TPW permita omitirlo, sino que permite ponerlo (en contra de las normas de Pascal estándar), aunque no sea necesario.

EL SALUDO DESDE EL MENÚ DE INICIO

Hasta ahora sólo se ha ejecutado desde dentro del entorno de desarrollo de TPW el primer programa que se ha escrito. A continuación, se estudiará la característica que ilustra las posibilidades que ofrece un lenguaje de programación de propósito general.

El programa de saludo que ha compilado TPW ha dado lugar a una aplicación que se puede ejecutar desde el sistema operativo Windows. Por ejemplo, se puede hacer que aparezca el saludo desde el menú de Inicio. Ahora se explicará cómo.

Acudir al menú de Inicio de la barra de tareas de Windows 95 y elegir la opción Ejecutar, para después accionar el botón Examinar. Así, se podrá comprobar que TPW ha producido un archivo ejecutable. Efectivamente, se trata del archivo saludo.exe y se encontrará en el mismo directorio donde se localiza el fichero que contiene el programa. En este caso, el directorio c:\tpw. Seleccionar este directorio en el árbol que presenta la ventana de la opción Examinar y comprobar que efectivamente se encuentra ahí el archivo.

Una vez elegido el programa, pasar a ejecutarlo. El resultado será que aparece la ventana que contiene el saludo.

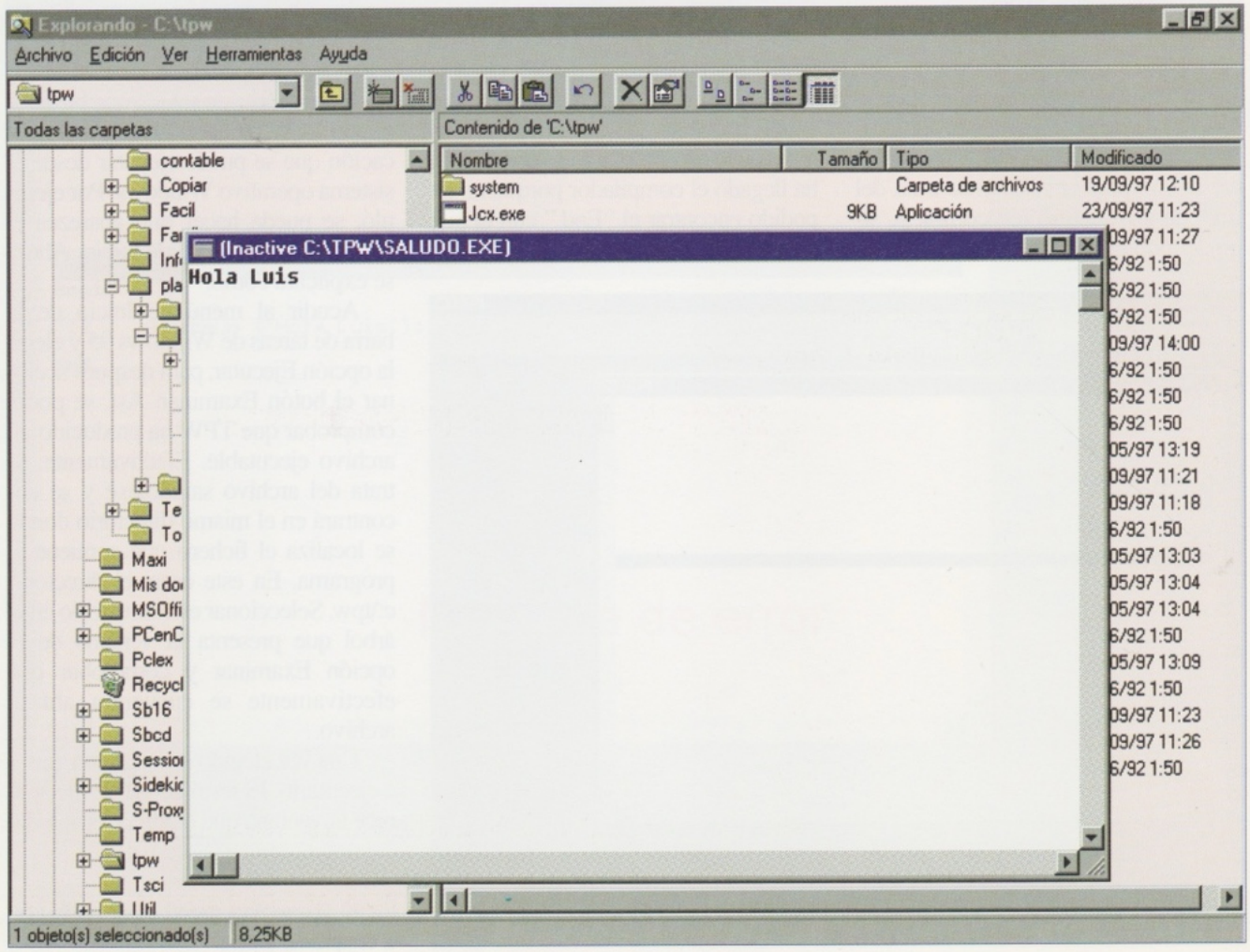
Fig. 20. El programa se acaba antes de lo que el compilador espera.

Fig. 21. El programa que realiza el saludo ha dado lugar a una aplicación más que se puede ejecutar desde el menú de Inicio.

Después de leerlo, se puede cerrar la ventana como si se tratara de cualquier otra aplicación pues, de lo contrario, como ya se señaló anteriormente, estará ocupando memoria durante el resto de la sesión de Windows.

Con todo esto se habrá podido comprobar que TPW pone al alcance de la mano la posibilidad de escribir aplicaciones que se integran de modo natural en el entorno de trabajo de Windows. A medida que se vaya dominando la sintaxis del lenguaje Pascal, podrán irse ampliando las posibilidades de las aplicaciones que se escriban.

Fig. 22. El programa que realiza el saludo ha dado lugar a una aplicación más que se puede ejecutar desde el menú de Inicio.



ENTRADA Y SALIDA DE DATOS

Ya se ha explicado con el programa de saludo que la función WriteLn sirve para que el programa Pascal se comunique con el usuario. Su nombre está formado por la contracción de Write Line (Escribe Línea). La palabra línea aparece en el nombre porque la función escribe sus argumentos en una línea. Es decir, que al terminar de escribirlos, envía el mensaje pertinente para cambiar a la línea siguiente.

Existe, claro está, la función inversa que permite al programa captar datos del usuario. Su nombre es ReadLn, y recuerda que la acción que ha de realizar es leer (*Read*, en inglés).

Para entender el uso de esta nueva función es necesario comprender el significado de las variables en los programas.

Con la función WriteLn en el programa de saludo, en la instrucción:

```
{1} WriteLn( ' Hola Luis ' ) ;
```

se está ordenando la escritura en el terminal de un dato. Ésta es la cadena de caracteres 'Hola Luis'.

Se trata de un dato que es el mismo cada vez que se ejecuta el programa, y que se guarda, junto al código de la UCP que ha de producir el saludo, en el archivo .exe que se creó con la compilación. Pero un programa puede hacer mucho más. Puede conseguir que sea variable, de una ejecución a otra, el dato que escribe en el saludo.

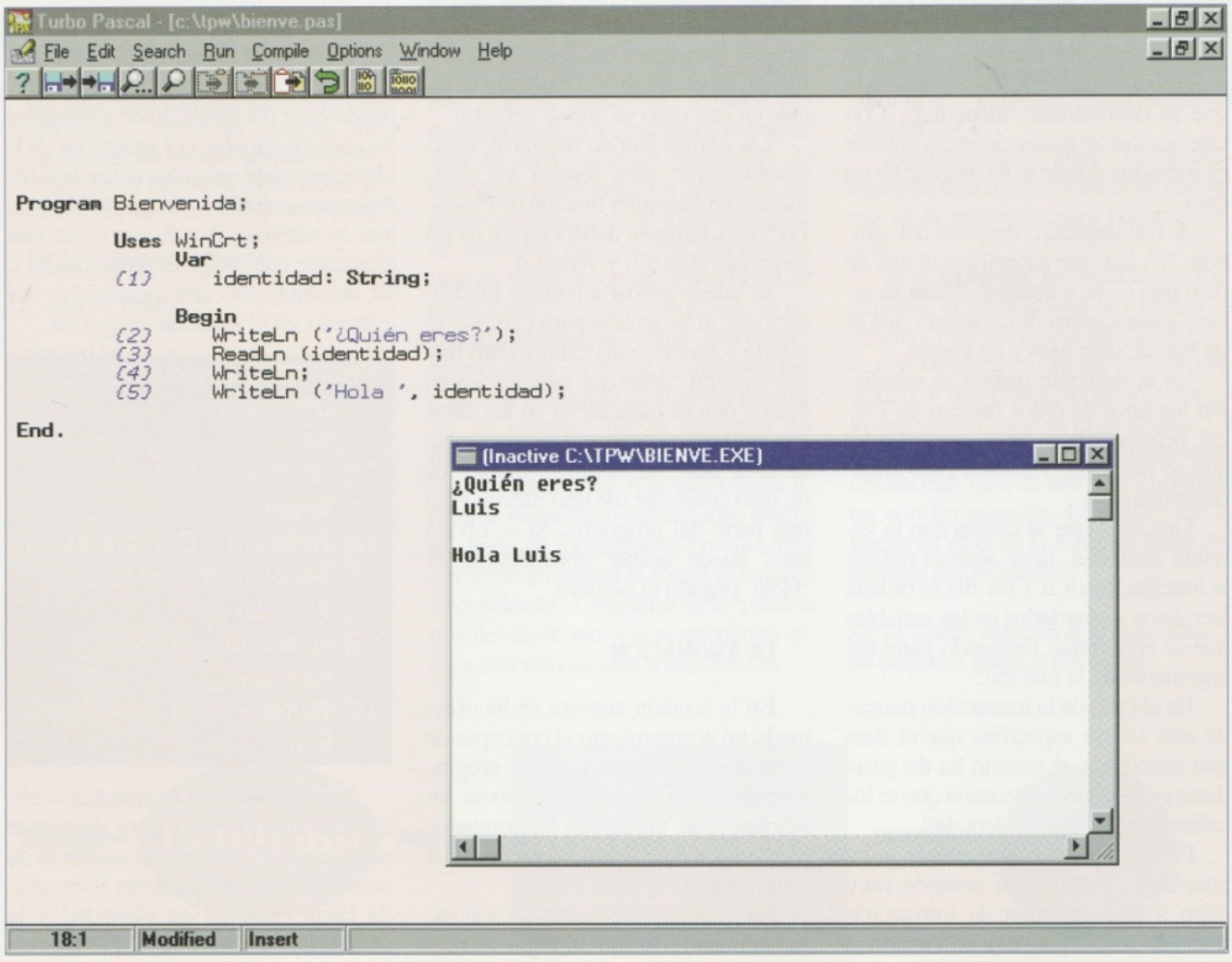
Suponer que se quiere ampliar el diálogo con el ordenador en la bienvenida, escribiendo un programa que pregunte el nombre del usuario y luego conteste con el saludo pertinente.

La secuencia de pasos que ha de realizar el programa se puede especificar así:

```
Preguntar el nombre.
Leer el nombre.
Escribir el saludo.
```

Se trata de la secuencia de instrucciones que ha de realizar el ordenador, escrita en un lenguaje muy próximo al natural. A esta especificación de las instrucciones del programa se le denomina pseudocódigo y es muy conveniente escribir el programa en él antes de hacerlo en el código del lenguaje que se está utilizando.

Fig. 23. Aspecto de la ventana de Windows que contiene el diálogo entre el programa y el usuario cuando se ejecuta el programa Bienvenida.



Llevado a un programa Pascal, este pseudocódigo podría ser así:

```
Program Bienvenida ;
    Uses WinCrt ;
    Var
{1}     identidad: String ;
Begin
{2} WriteLn( '¿Quién eres?' ) ;
{3} ReadLn( identidad ) ;
{4} WriteLn;
{5} WriteLn( 'Hola ', identidad ) ;
End.
```

Ya se vio anteriormente que después del nombre del programa se incluían las declaraciones, esto es, la información que necesita el compilador para interpretar correctamente las instrucciones que se encuentran en el cuerpo principal del programa (entre Begin y End). Con la palabra reservada Var se anuncia que lo que sigue son declaraciones de variables; para este programa se necesita una variable a la que se denominará "identidad". Con este nombre se hace referencia al lugar de memoria donde se ha de guardar un dato.

A continuación, después del símbolo ":", hay que especificar el tipo de dato que se va a guardar. String se refiere a una cadena de caracteres que es un tipo de dato básico en Pascal.

En la siguiente unidad se estudiarán los tipos de datos básicos de Pascal, pero por ahora basta con entender que se necesitan variables para almacenar datos.

Gracias a que se cuenta con la variable identidad, tiene sentido utilizar la función ReadLn. Con ella se ordena leer datos y guardarlos en las variables que se encuentran formando parte del argumento de la función.

En el caso de la instrucción marcada con {3} se especifica que el dato que introduzca el usuario ha de guardarse en el lugar de memoria que se localiza con el nombre identidad.

Puesto que está claro donde se ha guardado el dato, está también muy claro a qué se refiere la instrucción marcada con {5}: se está ordenando la escritura del dato (constante) Hola y, justo a continuación, el dato almacenado en identidad.

Utilizar TPW para escribir el programa Bienvenida y comprobar su funcionamiento. Hay que recordar que el nombre de un archivo no puede tener más de ocho caracteres.

Ya se habrá adivinado que la instrucción {4} es la que se encarga de dejar una línea en blanco entre la pregunta y el saludo. La función WriteLn, sin ningún argumento, manda a la salida solamente el mensaje de cambio de línea.

La función ReadLn siempre ha de tener un argumento variable, porque éste ha de ser un indicativo del lugar, o los lugares, donde se pueda guardar la información que se vaya a leer. TPW también permite utilizar ReadLn sin argumentos (pero sin los paréntesis que los delimitan) para indicar que el programa se interrumpe hasta que el usuario introduzca un dato. Evidentemente, es irrelevante el dato que se teclee en este caso ya que se perderá.

Las comas han de utilizarse, inexcusablemente, para separar los argumentos en cualquier función de Pascal. Por tanto, también deberá ser así en las funciones ReadLn y WriteLn.

Se puede probar a realizar modificaciones al programa para cambiar el saludo y hacerlo más cálido. Pero hay que tener en cuenta que los espacios en blanco que se especifican en los datos son también caracteres. Una secuencia de ellos entre las comillas simples es un dato constante (de tipo String) y no una parte del programa. Si se olvida esto, puede acabar obteniéndose el 'Hola' pegado al nombre.

LA ASIGNACIÓN

En la sección anterior se ha efectuado un acercamiento al concepto de variable, fundamental en la programación. A la variable se asocia un nombre que utiliza el programador para referirse a ella en la escritura del programa.

En la ejecución de aquél, cada variable tiene asociado un lugar de memoria donde puede almacenarse un dato. La mayoría de las órdenes que se escriben en un programa tienen como objetivo guardar un dato en la memoria del ordenador. Por ejemplo, la función ReadLn lee datos que suministra el usuario y los almacena en la memoria.

Una de las instrucciones básicas de la mayoría de los lenguajes de programación es la asignación. En Pascal se especifica utilizando el símbolo ":=" para indicar que se ha de asignar a la variable que se encuentra a su izquierda el dato que se localiza a la derecha. Por ejemplo, con

```
identidad := ' Antonio ' ;
```

se especifica que el dato (del tipo String) Antonio se almacene en la variable identidad. La orden acaba, como la mayoría, con el signo ; de fin de instrucción.

En la parte izquierda de la asignación debe encontrarse una variable, y ha de estar declarada anteriormente en la parte declarativa del programa. A la derecha puede encontrarse un dato del tipo que se declaró que puede almacenar la variable. Aquél puede ser una constante que indica el programador o el resultado de una operación. Por ejemplo, en el siguiente programa

```
Program Bienvenida2;
    Uses WinCrtt;
    Var
{1} identidad: String ;
    Begin
{2} WriteLn( '¿Quién eres?' ) ;
{3} ReadLn( identidad ) ;
{4} WriteLn ;
{5} identidad := 'ILUSTRE' +
    identidad ;
{6} WriteLn( 'Hola ',identidad ) ;
    End.
```

La asignación de la instrucción etiquetada con {5} sirve para almacenar en la variable identidad el resultado de unir la cadena de caracteres almacenada hasta entonces en identidad y la cadena de caracteres 'ILUSTRE'.

Con ello, el saludo que produce el programa será más amable, quizás demasiado. Ya se habrá comprendido que el signo +, situado entre cadenas de caracteres, da como resultado una nueva cadena formada por la unión de las dos (primero la situada a la izquierda y luego la de la derecha).

EL ESPAÑOL EN LA PROGRAMACIÓN

Pascal, como la mayoría de los lenguajes de programación de propósito general, se ha diseñado partiendo del inglés. Es por ello que sólo cuenta con las letras que aparecen en ese idioma. Tanto la Ñ como las vocales acentuadas y la u con diéresis son caracteres que no se consideran como posibles integrantes de los identificadores. Así, no es posible en un programa Pascal utilizar un nombre de variable en el que se incluya una Ñ o una vocal acentuada. Otra cosa distinta son los datos constantes, como las cadenas de caracteres, en las que pueden intervenir todos los de la tabla ASCII y, por tanto, también la Y y compañía.

No se puede usar la ñ en

```
año := 'Enero' ;
```

pero sí en

```
Enero := ' Año 1995 ' ;
```

En el primer caso se está intentando asignar un valor a una variable que utiliza la ñ en el identificador.

En la siguiente unidad se estudiarán los distintos tipos de datos definidos en lenguaje Pascal y los operadores que permiten manipularlos.

GRÁFICOS SINTÁCTICOS

Se habrá preguntado cómo analiza sintácticamente el compilador de TP los programas de Pascal. También habrá pensado que si el ordenador es capaz de decidir si un programa es sintácticamente correcto, será porque alguien ha programado un algoritmo que es capaz de realizar la verificación.

El compilador tiene que determinar si el programa cumple todas las reglas sintácticas en la fase de verificación. Para ello, va desmenuzando aquél de lo general a lo particular, averiguando si se quebranta alguna de las reglas del lenguaje. Si fuera un texto, se diría que analiza el párrafo y lo desmenuza en oraciones. Luego, cada una de las oraciones las analiza en términos de las palabras, verificando que éstas forman las oraciones, y varias oraciones los párrafos, siguiendo las reglas del idioma.

Es muy útil considerar los llamados gráficos sintácticos, también denominados grafos sintácticos o diagramas en carril (railroad diagrams). Se trata de la representación gráfica de las reglas sintácticas de un modo que puede utilizarse para escribir los algoritmos de verificación sintáctica (y los de traducción a lenguaje máquina). Se recorren siguiendo el "carril" que constituye la línea, en la que se incluyen flechas que indican el sentido del recorrido. Se empieza por la izquierda del diagrama y se sigue en la dirección que marcan las flechas. A lo largo de las líneas se encuentran enmarcados distintos tipos de indicadores. En ellos hay que tener en cuenta la forma del marco (redondeado o cuadrado), la palabra que llevan dentro, y si se encuentra escrita con mayúsculas o minúsculas.

Una parte del programa es correcta sintácticamente si empezando por la izquierda del gráfico se consigue, siguiendo el sentido indicado, salir del gráfico, cumpliéndose que todas las palabras o signos con los que se tropiece en el programa corresponden a los indicadores del gráfico.

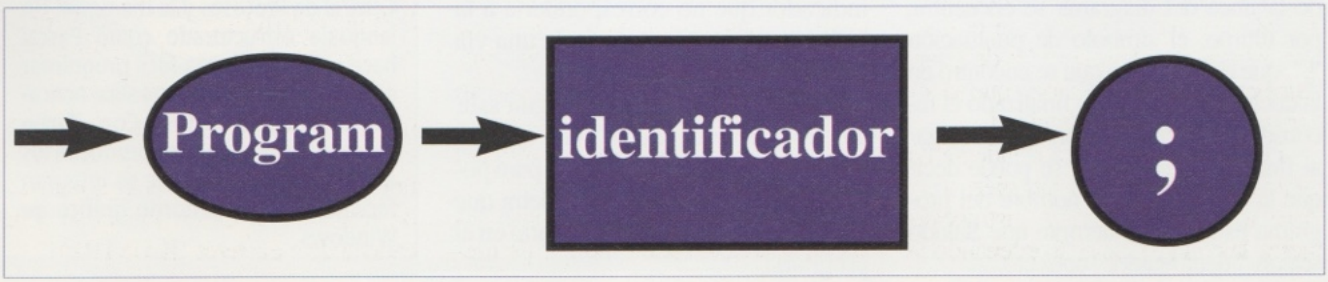
Un indicador redondeado contiene un símbolo, un signo de puntuación o una palabra. Si ésta aparece escrita en mayúsculas quiere decir que se trata de una palabra reservada de Pascal y que la que se encuentre en la frase analizada tiene que coincidir exactamente, salvo mayúsculas o minúsculas, con la que se encuentre en el diagrama.

Cuando en el indicador redondeado aparece una palabra en minúscula es que su significado es obvio y lo que se encuentre en el programa ha de estar de acuerdo con el significado. Por ejemplo, si aparece la palabra "dígito" querrá decir que en el programa se encontrará alguno de los dígitos situados entre el 0 y el 9 para poder seguir el camino.

En el caso de los signos de puntuación está claro que se habrán de encontrar en el programa para poder continuar.

Si el indicador es cuadrado o rectangular, el nombre que hay dentro se refiere a otra estructura de Pascal para la que también existe un gráfico sintáctico, y que puede verificarse con él lo que se encuentre en el programa. La idea que hay detrás de estos diagramas resulta más fácil de entender si se ilustra con un ejemplo. El gráfico de la figura 24 es el de la especificación del nombre del programa.

Fig. 24. Gráfico sintáctico de la declaración del programa, quizás el más simple.



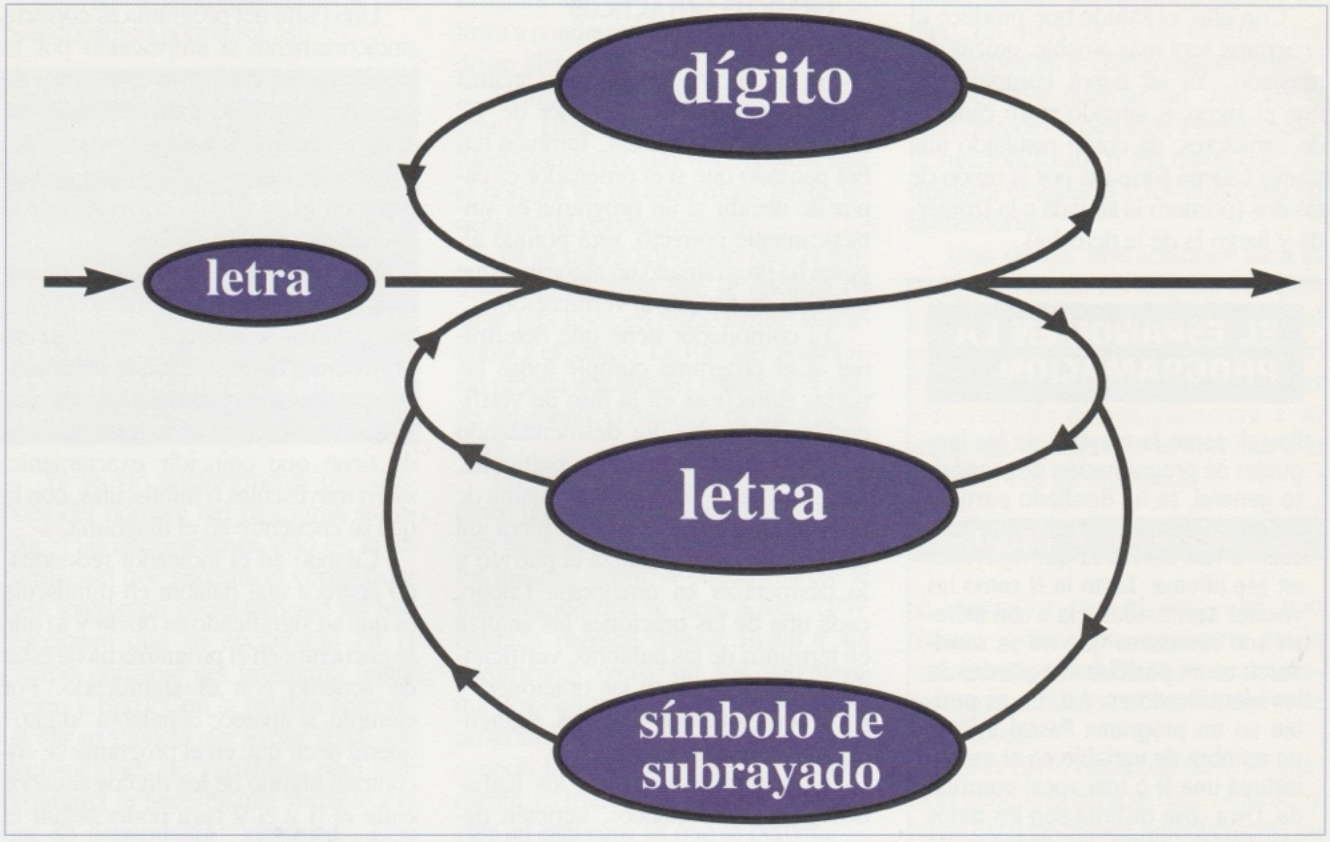


Fig. 25. Gráfico sintáctico de los identificadores que se utilizan para dar nombres a programas, constantes, variables, procedimientos, y otros.

Se va a verificar si

```
Program Saludo;
```

es una declaración correcta de nombre de programa. Hay que empezar el recorrido del diagrama, y al mismo tiempo de la declaración, por la izquierda. Un lápiz puede resultar útil como puntero para efectuar este recorrido. En el diagrama se encuentra la palabra reservada Program que también aparece en la declaración. Por ahora todo es correcto y se puede salir del indicador para seguir el recorrido. Continuando por la línea del diagrama se encuentra "identificador" y, a su vez, en la declaración aparece Saludo. Se deja para verificar después que Saludo es un "identificador" correcto. Continuando el seguimiento de la línea del diagrama se encuentra, por último, el símbolo de puntuación ";", que es lo mismo que se encontró en la declaración. Ya se ha finalizado el recorrido, también se ha conseguido llegar al final del diagrama, y se puede decir que la declaración del nombre del programa es correcta, siempre que Saludo sea un "identificador" correcto.

Hay que acabar la verificación de la declaración del programa Saludo; cumple con la reglas sintácticas de "identificador": empieza por una letra y le siguen también letras.

En el diagrama se empieza por la izquierda y, efectivamente, primero aparece la letra "S". A continuación, en el diagrama hay una bifurcación que no se puede tomar porque se circularía en dirección prohibida.

Después, hay otra que, esta vez sí se puede tomar en el sentido que se está recorriendo el diagrama. Hay que tomar la curva adecuada, hacia abajo, para tropezar con el indicador de la letra que satisface la "a" de Saludo. Si se hubiera tomado el camino del dígito, se habría tropezado con un indicador que no correspondería a la "a" y, por tanto, se trataría de una vía muerta.

Como el objetivo es conseguir salir del "diagrama de carril", hay que seguir dando vueltas hacia abajo para poder pasar por el indicador de letra que corresponde a lo que se encuentra en el programa.

Unas cuantas vueltas más y se sale del diagrama verificando que el identificador es correcto.

Como ejercicio, le proponemos comprobar que 1deTantos no es un identificador correcto y que tampoco lo es la declaración de programa

```
Program Buenos Dias ;
```

RESUMEN

Programar el ordenador es dirigir el tráfico de datos entre la Unidad Central de Procesos y la memoria. Un lenguaje estructurado como Pascal hace posible que sea fácil programar el ordenador para que realice aplicaciones muy diversas. Con Turbo Pascal para Windows se producen los archivos ejecutables que se integran fácilmente en el entorno gráfico de Windows.

CUESTIONARIO AUTOEVALUATIVO

1 - El trabajo que realiza el ordenador se puede resumir con la frase:

- ❑ A - La velocidad de proceso de la memoria.
- ❑ B - La capacidad de almacenamiento de la CPU.
- ❑ C - El tráfico de datos dirigido por la CPU entre el exterior y la memoria.
- ❑ D - Las operaciones matemáticas de la ALU.

2 - Un programa Pascal en el que aparece la expresión z := x + y ; está:

- ❑ A - Igualando la z a la suma de x e y.
- ❑ B - Calculando la suma de z, x e y.
- ❑ C - Calculando la suma de x e y.
- ❑ D - Calculando la suma de los datos almacenados en las variables x e y.

3 - ¿Qué es un compilador?

- ❑ A - Un lenguaje de bajo nivel que comprende la UCP del ordenador.
- ❑ B - Un programa capaz de hacer pilas de objetos.
- ❑ C - Un lenguaje primitivo de alto nivel que ya no se utiliza.
- ❑ D - Un programa que convierte el código de un programa en instrucciones entendibles por la UCP.

4 - ¿Cuál de los siguientes es un principio correcto en un programa Pascal?

- ❑ A - Begin Program Uno;
- ❑ B - Program Pascal;
- ❑ C - Programa Uno;
- ❑ D - Pascal Program;

5 - Todo comentario correcto de Pascal que empieza por { acaba con:

- ❑ A - }
- ❑ B - *)
- ❑ C - (*
- ❑ D - {

6 - ¿Cuál de los siguientes nombres es válido como indicador de una variable?

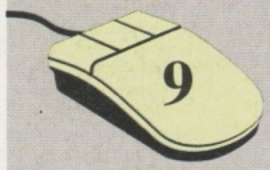
- ❑ A - Año.
- ❑ B - NigNO.
- ❑ C - Sería.
- ❑ D - Program.

7 - ¿Cuál de las siguientes afirmaciones es falsa?

- ❑ A - Un programa compilado con TPW se puede ejecutar desde el entorno de desarrollo de TPW.
- ❑ B - Un programa compilado con TPW se puede ejecutar desde el menú de Inicio de Windows 95.
- ❑ C - Un programa compilado con TPW se puede ejecutar desde el Explorador de Windows 95.
- ❑ D - Un programa compilado con TPW sólo se puede ejecutar desde el icono de TPW.

8 - Una de las siguientes sentencias es siempre incorrecta en un programa Pascal. ¿Cuál es?

- ❑ A - WriteLn(' Año ');
- ❑ B - Readln;
- ❑ C - Readline(Una, ' Otra ');
- ❑ D - WriteLn(Una, ' Otra ');



9 - El programa de Pascal program uno ; begin readln ; writeln(' adiós ') end.

- ❑ A - Daría un error en compilación por el uso indebido de la mezcla de mayúsculas y minúsculas.
- ❑ B - Daría un error en compilación por estar escrito en una sola línea.
- ❑ C - Daría un error en ejecución porque realiza una lectura fallida.
- ❑ D - No produciría ningún mensaje de error ni en compilación ni en ejecución.

10 - La instrucción Uses WinCrt ;

- ❑ A - Debe incluirse al final de la parte declarativa de los programas.
- ❑ B - Tiene que incluirse antes de cualquier otra instrucción.
- ❑ C - No puede ir acompañando a una declaración de variable interna.
- ❑ D - Declara que se utilizará la unidad externa que permite realizar el intercambio de datos en ventanas de Windows.

Respuestas: 1 C - 2 D - 3 D - 4 B - 5 A - 6 B - 7 D - 8 C - 9 D - 10 D